(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 January 2001 (04.01.2001)

PCT

(10) International Publication Number
WO 01/01297 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US00/18183

(22) International Filing Date: 29 June 2000 (29.06.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

09/345,225	30 June 1999 (30.06.1999)	US
09/345,259	30 June 1999 (30.06.1999)	US
09/345,170	30 June 1999 (30.06.1999)	US

(71) Applicant: BIZTRO, INC. [US/US]; 2500 Augustine Drive, Suite 100, Santa Clara, CA 95054 (US).

(72) Inventor: D'SOUZA, Roy, P.; 657 Spruce Drive, Sunnyvale, CA 94806 (US).

(74) Agent: VILLENEUVE, Joseph, M.; Beyer Weaver & Thomas, LLP, P.O. Box 130, Mountain View, CA 94042-0130 (US).

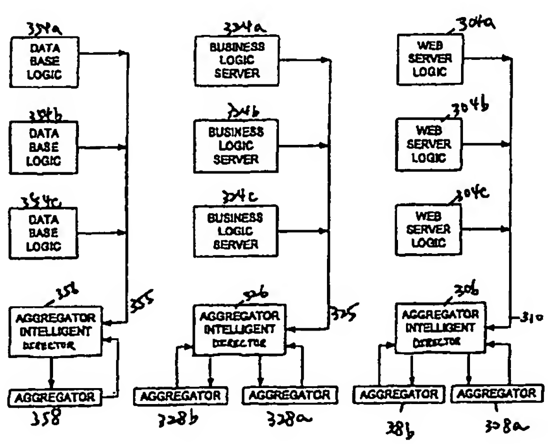
(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: DATA AGGREGATOR ARCHITECTURE FOR DATA MINING



(57) Abstract: A computer infrastructure relating to strategies for more efficient and effective data mining includes clusters of web servers (304), business logic servers (324a, 324b, 324c), and database servers (354a, 354b, 354c). Each server cluster is linked to a corresponding functional intelligent director (355, 326, 306) that receives and responds to access requests by assigning and possibly reconfiguring the servers (304). Each server (304) is also linked to an aggregator network, whose complexity may vary from one aggregator to a complicated network of aggregator intelligent directors (355, 326, 306) and aggregators. The complexity of each aggregator network may vary depending on the data mining strategy and the needs of the existing infrastructure. Events determined to be suitable for data mining are created, followed by setting up listeners, and providing handlers to capture generated data. An event is turned on or off using flags in an event table.

WO 01/01297 A1

DATA AGGREGATOR ARCHITECTURE FOR DATA MINING

FIELD OF THE INVENTION

The present invention relates to uses of a computer infrastructure and more particularly, to improved methods of implementing data mining processes utilizing a number of novel approaches to achieve various purposes.

BACKGROUND OF THE INVENTION

With the rapid proliferation of computers and the vast quantities of information they can provide, various techniques are used to capture this data and to sort through it. One technique is known as knowledge discovery, and is often referred to as "data mining" in the popular press. Data mining is unlike a traditional well-organized and robust database that a user might use to formulate a specific database query. Data mining refers to situations that have three major characteristics.

Firstly, there is an incredible amount of data that may be difficult to sort through or may be coming from multiple databases. Furthermore, this may be a large amount of static data, or it may be data that is being captured and stored quite rapidly. Secondly, the data itself might be suspect. In other words, unlike a traditional well-organized database, the data that is the subject of data mining might not be accurate, may have errors, or may not quite be what one expects. Thirdly, and most importantly, in a data mining situation the user is not quite sure what to ask for in terms of a database query. In fact, the user may not be trying to formulate a query, but rather, is trying to analyze the data to see if any patterns exist. In other words, the user is trying to determine patterns or secrets that the data may hold without knowing what they are beforehand. In a nutshell then, data mining refers to a situation where the user is looking for information in the data but is not sure what that information might be.

One way to capture the data in a computer system or a computer network for future data mining uses the programming technique of raising events throughout the software code. Raising events, or more appropriately, making a "raise event" call, is a

well known programming technique of capturing data at a particular point in the software code and outputting it to a computer screen, disk or other storage device. For example, raise event calls are used frequently in the debugging of software to output variable values at particular points in the code so that the code may be debugged. In another example, a typical business application software program might have raise event calls sprinkled liberally through the code in order to capture large volumes of any and all type of data.

In an ideal situation, prior art data mining techniques would determine what type of data is to be collected at the onset, so that raise event calls which are used to indicate the occurrence of specified activities may be written into the business application code as the code is being created. This requires a great deal of foresight in determining what types of data would be useful for data mining and may result in the collection of either too much or too little data, which would either saturate the disk and processor capacity or would provide inadequate information for data mining purposes. Moreover, it is difficult to predict what type of data to collect from information generated on the Internet. Simply placing the raise event calls everywhere in the code would generate too much data.

Business applications are generally not designed with data mining capabilities in mind, thus, an effort must subsequently be made to go into the application code and graft the necessary raise event calls into the business application after the code has been written. This requires that a programmer understand the business application as well as retrofit the code to include the raise event calls to collect the desired information. Moreover, once these raise event calls are placed into the code, they will continue to collect the specified information until a conscious effort is made to modify the code and cancel the raise event call (either by making them a comment, or by deleting them altogether). After modification, the code must be recompiled to enable it to operate. Adding new raise event calls requires a conscious effort to modify the code to incorporate the new raise event calls and to recompile the code to make it functional. Therefore, what is desired would be a data generating and storage system that allows flexibility in the capture and storage of data for data mining. Further, such a system might function independently of thc business application code so as to eliminate redundant procedures when the logistics of data collecting and data mining are changed.

Another issue that arises in using prior art techniques is the storage of data in a single machine, such as in a message queue. When this machine is saturated with data, it must be replaced in its entirety with a larger machine to avoid slowing down the processor or deluging the hard disk with too much data. The disadvantages of this approach is that it lacks expansion flexibility needed to accommodate increasing numbers of users as well as increasing amounts of data. Therefore, what is needed is a storage arrangement that would provide a feasible scaling solution to allow expansion as necessary to avoid bringing about a bottleneck in any part of the operation.

Figure 1 illustrates a prior art computing infrastructure for use by users accessing a target web site. In the prior art infrastructure, a user puts in an access request 102 that goes through a web server router 104 which is linked to a plurality of web servers 106a-e. Web server router 104 randomly assigns a web server among web servers 106a-e to respond to the user's access request 102. Web servers 106a-e are linked to an application server router 110, which assigns an application server from the plurality of application servers 112a-d. Each of these application servers 112a-d is also linked to a database router 120, which in turn is linked to databases 124a-c. Database router 120 may assign a database among databases 124a-c in response to an access request as needed.

As shown in the above example, each server (which may be a database server, an application server or a web server), is connected to at least one router, which plays the intermediate role of assigning the appropriate server in response to an access request. Each router is generally linked to like servers in a given cluster, and selects one of these servers for assignment based on factors such as machine load and existence of previously cached data. Information for data mining is generated during normal operation of these servers, which means that the data gathering is tied into the business application code, and that the code will have to be modified and recompiled when changes to the data gathering strategies need to be made. Moreover, the data generated is stored within this infrastructure, which does not allow for expansion in terms of data storage capability. Therefore, what is needed is an improved expandable computer infrastructure that will allow for a smooth extension of data storage capabilities as well as improved schemes that will allow the business

applications and the gathering of generated data to function independently of one another.

SUMMARY OF THE INVENTION

To achieve the foregoing and according to the purpose of the present invention, improved computer infrastructures as well as improved techniques for capturing data for data mining are disclosed.

In a first embodiment of the invention, a method of creating an event for a user activity is disclosed. The event is raised when its corresponding user activity occurs. This method includes determining that a user activity is useful for data mining, creating an event for this user activity, assigning an event name to the created event, recording the event name in an event table and setting the event to a default state, placing calls in appropriate locations in computer code to raise the event when its corresponding user activity occurs, and compiling the computer code to allow each event to be raised when its corresponding user activity occurs. Optionally, this method may further include setting up a listener to listen for events selected from a list of documented events. The setting up of a listener would include the steps of reviewing the list of documented events, selecting at least one event for a particular business need, and writing handler code for each selected event which would handle that particular selected event should it be raised.

In another aspect of the first embodiment, a method of processing an event is disclosed. This method includes receiving an indication of an event when that event is raised, obtaining an event identifier for that event, looking up the identifier in an event table to determine if the event is turned on, and if the event is turned on, a handler for that event is invoked and run. As the handler runs, it may also capture data to an aggregator. Alternatively, the event processing may include the signaling of a daemon thread.

In yet another aspect of the first embodiment, a server used within this computer infrastructure is disclosed. The server includes a code module, a handler module, a filtering/policy module and an event processor. Possible variations in the server structure may include a filter/policy module in the server having an event table with a plurality of events or a code module having a plurality of raise event calls embedded in the code.

This first embodiment advantageously provides for a dynamic selection of the type of information that is captured to look for patterns and other information through the use of dynamic raise event calls. By way of example, the dynamic events may be embedded in the code and can be turned on or off depending on what information is to be collected for data mining.

In a second embodiment of the invention, an aggregator intelligent director computer is disclosed. This aggregator intelligent director computer includes a processing capability database, a load status database, an aggregator selector and a configurator. The aggregator intelligent director computer may optionally include a usage history database.

In another aspect of the second embodiment, an aggregator network is disclosed. The aggregator network includes an aggregator intelligent director linked to a number of aggregators. The aggregator network may also be further expanded by linking each of the aggregators to another aggregator intelligent director, which in turn is linked to a second set of aggregators.

In yet another aspect of the second embodiment, a computer network is disclosed. This computer network includes a number of web servers, a number of business logic servers, a number of database servers and an aggregator network. In a variation on this embodiment the aggregator network includes an aggregator intelligent director.

In a further aspect of the second embodiment, a method for capturing data is disclosed. This method for capturing data includes receiving data from a server, checking databases in an aggregator intelligent director computer, selecting an aggregator using an aggregator intelligent director computer, and delivering the data to the aggregator. The method may further include accepting feedback from the aggregator and including that feedback in the databases. Alternatively, the method may include configuration of the aggregator.

The second embodiment advantageously provides an improved aggregate architecture for storage of captured data for data mining that allows virtually unlimited expansion in capacity by adding machines as needed. This advantageously eliminates the need to replace machines already present within the infrastructure.

In a third embodiment of the invention, a policy module that is decoupled from a primary business application is disclosed. This policy module includes an event table having a number of events and a toggling switch to toggle the events on or off. The policy module may further include a counter to increment occurrences of each of the events. Alternatively, the toggling switch in the policy module may toggle the events on or off independent of a main business application that is running alongside the policy module.

The third embodiment provides for decoupling the policy module from the primary business application. This advantageously allows a policy module to be created, modified and run independently of operations relating to the primary business application. This policy within the policy module may be modified as needed to fulfill data mining requirements by toggling events on and off as needed. This approach avoids recompiling of code and allows for tailoring of policies without affecting normal operation of the primary business operation, therefore resulting in greater efficiency.

Other aspects and advantages of the invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrating by way of example the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings. In the following figures, like reference numerals refer to analogous or similar elements to facilitate ease of understanding.

Figure 1 illustrates the prior art computing infrastructure for use by users accessing a target web site.

Figure 2 illustrates an improved computing infrastructure for use by users accessing a target web site having greater expansion capabilities through the use of intelligent directors.

Figure 3 shows a simplified illustration of the computing infrastructure of Figure 2 having a number of aggregator networks connected to the servers for information downloading.

Figure 4 illustrates an exemplary aggregator computer network having aggregator intelligent directors and aggregator computers forming a tree topology.

Figure 5 illustrates components of an exemplary aggregator intelligent director.

Figure 6 illustrates an exemplary server computer having a code module, a filtering/policy module, a handler module and an event processor.

Figure 7 illustrates a process flow of creating an event for use in data mining.

Figure 8 illustrates a process flow of setting up a listener for capturing selected events.

Figure 9 illustrates a process flow of processing an event once that event is raised.

Figure 10 illustrates a process flow to filter the collecting of data by taking actions in response to events raised.

Figures 11 and 12 illustrate a computer system 900 suitable for implementing embodiments of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will now be described in detail with reference to the preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order to not unnecessarily obscure the present invention.

The first embodiment relates to an improved method that allows for dynamic selection of the type of information that is captured to look for patterns and other information through the use of dynamic raise event calls. According to this embodiment, events that have been created by placing dynamic raise event calls in the code may be turned on and off depending on what information needs to be collected. This allows for flexibility in modifying data mining policies; each policy may be

tailored to the proper problem domain without going through extensive procedures such as modifying and recompiling the main code to achieve that purpose.

The second embodiment relates to an improved aggregate architecture to allow for flexible expansion of the storage capacity for collecting data for data mining purposes. According to this embodiment, an aggregate architecture may start with a basic topology and be expanded by adding additional aggregate intelligent directors and aggregator computers to increase storage capacity. Conversely, the topology may be modified by removing aggregator intelligent directors and aggregator computers when they are not needed.

The third embodiment relates to an innovative feature of decoupling the filtering/policy module from the primary business application. The decoupling feature advantageously allows data collection to take place without interference from the operations of the primary business application. According to this invention, the filtering/policy module is a separate entity from the code module that handles the main sequence of operations. Altering the filtering/policy module by turning selected events on and off will have the effect of modifying the data collection strategy without affecting the primary structure of the business code module.

COMPUTING AND DATA CAPTURE ARCHITECTURE

Figure 2 illustrates an improved computing infrastructure for use by users accessing a target web site, wherein the routers that are used in the prior art infrastructure to direct users to their assigned servers are replaced by intelligent directors. In this example, when a user puts in an access request 102 the request goes through a web logic intelligent director 204 which is linked to a plurality of web servers 106a-e. Web logic intelligent director 204 receives feedback from web servers 106a-e, and based on the feedback, selects a web server among web servers 106a-e to respond to the user's access request 102. Web servers 106a-e, which are linked to a business logic intelligent director 210, have the capability to request a business logic server based on the user's needs via business logic intelligent director 210. Director 210 selects a business logic server from the plurality of business logic servers 112a-d based on the information it receives from the requesting web server and feedback from the business logic servers. Each of these business logic servers 112a-d is also linked to a database intelligent director 220, which in turn is linked to

databases 124a-c. This enables a business logic server to select a database via database intelligent director 220. Like the other intelligent directors, database intelligent director 220 selects a database based on the requesting server, which is a business logic server in this case, as well as feedback from the databases it is linked to.

As shown in above example, each server, which may be a database server, a business logic server, or a web server, is connected to at least one intelligent director, which, among other functions, plays the intermediate role of assigning the appropriate server in response to an access request. The pair of lines pointing in opposite directions between any two machines illustrate the logical view of the connections that exist between these machines. However, in reality, the connection is enabled by a network interface card (NIC) and a bus having bidirectional functionality to allow input and output. If a machine is linked to two separate machines, each link has its own connection, each of which is enabled by its own network interface card and bus.

An intelligent director is generally linked to like servers in a given cluster and selects one of these servers for assignment based on feedback received from these servers. The use of intelligent directors in the improved computing infrastructure allows for greater expansion capabilities which enables the system to be readily adapted to meet the needs of an expanding base of users. A more detailed explanation of the functions of intelligent directors is provided in related U. S. patent application No.______________ (attorney docket number BHUBP008), which is incorporated by reference. Those of skill in the art will appreciate that use of an intelligent director is optional for implementing the present invention, although it is preferable.

The expansion capabilities provided by these intelligent directors may also be applied to the use of aggregator computers to expand storage capability for use in data mining, as will now be explained in Figures 3-5.

Figure 3 shows a simplified illustration of the computing infrastructure of Figure 2 having a number of aggregator computer networks connected to the servers for information downloading. Web servers 304a-c are linked to an aggregator intelligent director 306 which serves as a foundation for each aggregator computer network. Aggregator intelligent director 306 is, in turn, linked to aggregator computers 308a-b. Intelligent directors other than the aggregator intelligent directors

are not shown in this figure for ease of illustration. As the web servers process data and events, information is collected. This information may reach aggregator intelligent director 306 via network connection 310 wherein an aggregator computer may be assigned to receive that information. Information collected at business logic servers 324a-c and database servers 354a-c may be processed in a similar fashion via paths 325 and 355 to aggregator intelligent directors 326 and 356. These directors will assign an aggregator, for example, 328a, 328b or 358. Information collected in this manner may be used for data mining either in an offline mode where data is collected for a long period of time and then processed later to look for patterns, or alternatively, in a real-time mode where the data is dynamically processed as it is being collected and decisions are made on the spot or with a very short lag time.

For the purpose of downloading information for future use, each server of a cluster in this exemplary infrastructure is connected to an aggregator intelligent director, which may in turn be connected to an aggregator computer network which may include a single aggregator computer or multiple aggregator computers. These aggregators may stand alone, or alternatively, they may be expanded further through the use of additional aggregator intelligent directors into a hierarchy of aggregators. Aggregators and aggregator intelligent directors may be linked in a variety of combinations to form different aggregator networks having different topologies, depended on what is needed. For example, one topology might have all aggregators linked to one aggregator intelligent director, while another topology might have each aggregator linked to an individual aggregator intelligent director.

Figure 4 illustrates an exemplary aggregator computer network having aggregator intelligent directors and aggregator computers that form an exemplary tree topology. In this instance, aggregator intelligent director 400 is linked to multiple aggregator computers 410a-d. Aggregators 410b-d are expanded by linking to aggregator intelligent director 420, which in turn is linked to two aggregators 430a-b. Aggregator 430b is then linked to aggregator intelligent director 440, which in turn is linked aggregators 450a-b. This chain of aggregator intelligent directors and aggregators may continue as long as it is deemed reasonable and necessary. Assuming a request to download information is sent through aggregator intelligent director 400, aggregator 410c may be assigned to respond to that request. If aggregator 410c does not have adequate capacity to store all the information provided,

aggregator 410c may send a request to aggregator intelligent director 420, which may select aggregator 430b based on the nature of the request and feedback from its downstream aggregators, which in this case would be 430a-b. Aggregator 430b would continue to receive the information, but in the event that aggregator 430b has inadequate capacity, it may request additional reinforcement through aggregator intelligent director 440, which may assign an aggregator 450a or 450b depending on what is needed as well as the feedback it receives from these aggregators.

AGGREGATOR INTELLIGENT DIRECTOR

Figure 5 illustrates components of an exemplary aggregator intelligent director 500. The components of aggregator intelligent director 500 include an aggregator selector 502, a usage history database 504, a processing capability database 506, a load status database 508 and a configurator 510. Also included but not shown is an aggregator directory, a latency database, a log file database and a network traffic database. In this example, aggregator intelligent director 500 receives a request for an aggregator at 512 over a network connection from one of the web servers, business logic servers or a database server. Upon receipt of this request, aggregator selector 502 looks to usage history database 504, processing capability database 506, load status database 508 and information from the other databases to determine which aggregator can best be assigned to handle this particular request. These databases store status information about the aggregators and are linked to the aggregator intelligent director and may be updated by receiving feedback from these aggregators as shown in 514. Upon receipt of the request, the aggregator selector selects an aggregator computer based upon these databases and assigns an aggregator to handle the request and returns the aggregator selection in 516. Besides providing information to the aggregator selector, the databases also provide information to configurator 510 which may reconfigure the aggregators over connection 518 using the feedback information provided. Especially if the aggregators are heterogeneous, for example, certain aggregators may have special data mining or data gathering functions that other aggregators do not have.

Information received from the aggregators via path 514 is stored in exemplary blocks 504, 506, 508 and in the other databases not shown. For ease of illustration, not every database is shown in Figure 5. Note that some of information is static and

may be received as part of the registration process that the aggregators undergo as they are installed into the network. An example of such static information is aggregator processing capability and geographic location. Other information may be dynamically received by the director from the aggregators. Still, other information may be derived from the information received dynamically and/or statically (such as the average latency time for aggregators, which may be calculated periodically based on average network latency between the server and an aggregator).

Usage history database 504 is used to predict usage demands placed on the aggregators and to prospectively balance the data to capture among the aggregators if needed. Thus, an anticipated surge in usage does not overwhelm any particular aggregator. For example, if one particular aggregator is always overwhelmed each Saturday evening, that information is kept in database 504 and can be used in the future to deflect a sudden influx of data from that aggregator on a future Saturday evening.

Processing capability database 506 may track the processing capability (again in number of transactions per second, the number servers that can be supported concurrently, etc.) of the individual aggregators in order to ascertain how much bandwidth a particular aggregator may have, how much has been used, and how much is available. Information pertaining to the processing capability of the aggregators may powerfully impact the routing decision since a more powerful aggregator may be able to capture data faster than a less powerful aggregator even if the more powerful aggregator may appear to be more heavily loaded. The aggregator processing capability may be obtained when the aggregator is first installed and registered within the system.

Load status database 508 may track information pertaining to the level of load currently experienced by the individual aggregators (in number of transactions per second, the number of servers currently using an aggregator, CPU load, for example). This information may be tracked for aggregators individually and/or as a group average.

Network traffic data pertaining to specific connections within and between the various sites may be obtained through the use of appropriate network sniffers or other

software tools and furnished to a network traffic database (not shown) within director 500 so that the appropriate calculations pertaining to average latency can be made.

An aggregator average latency database (not shown) may track the average latency to be expected if a particular aggregator is employed to service a data capture request. The average latency may be calculated based on the processing capability of the aggregator, how remote it is from the server that issues the capture request and the speed of the network connection.

An aggregator log file database (not shown) may track the operational status of each aggregator to determine, for example, the length of time that the aggregator has been in operation without failure and other types of log file data.

An aggregator directory (not shown) may track information pertaining to the list of aggregators available to the system, their remote/local status (geographic distribution), their relative weight which reflects the relative preference with which data should be directed to the individual aggregators (e.g., due to network conditions or other factors), and the like.

Regarding the remote/local status of the aggregators, their geographic distribution may also impact routing decisions. Nowadays, it is common to find servers of a given service widely dispersed over a large geographic area, both to improve service to its worldwide customer base and also to provide some measure of protection against natural or man-made catastrophes that may impact a given geographic location. The aggregators that capture data from these servers may also be well dispersed.

In general, it is desired that data capture requests originated from a given locality be serviced by aggregators that are closest to the place of origination of the data. There are times, however, when local processing resources may not be adequate and remote aggregators need to be employed in order to reduce the capture request processing times. Further, if the local resources that the local servers need to service data capture requests are temporarily unavailable or overloaded, the delay may be less if remote aggregators are called upon to service the requests originated locally, particularly if the remote aggregator have ready and quick access to the needed database resource at the remote site. Still further, since the aggregators may be

interconnected with one another and with other components of the system using networking technologies, network congestion at specific locations may unduly increase the time required to process a capture request. Due to network latency, it may be possible to reduce the capture request service time by routing the transaction request to a remote aggregator for servicing.

When one of the servers prepares to capture data for later data mining (for example, as in step 920), it accesses selector 502 to ascertain the appropriate aggregator computer to which the data may be sent. The decision pertaining to which aggregator to assign the data may be made based on the current relative load levels on the aggregators, their history of usage, their processing capabilities or other. This information is periodically received by director 500 from the aggregators through path 514. Using the databases available, an aggregator selector 502 then selects one of the aggregators to service the pending data capture request and transmits the selection to the requesting server via path 516. Aggregator selector 502 may select an aggregator for data capture using any of a wide variety of methodologies. Indeed, the heuristics are limitless and may depend upon the nature of the aggregators, the data to be captured, and the actual implementation of the overall system. The information contained within databases 504-508 and those not shown may be used in many different ways to choose an aggregator for data capture. A few techniques will now be discussed.

With reference to Figure 5, for example, selector 502 may route the incoming data capture request to one of the aggregators using a conventional routing methodology (such as round robin, based on the relative load levels, and the like). In one embodiment, since director 500 has available to it performance data pertaining to the aggregators (such as processing capability, load status, latency, etc.), selector 502 may intelligently route the incoming data capture request to a specific aggregator that can service the request in the most timely manner. Additionally and/or alternatively, all aggregators may be loaded with data capture requests gradually and equally so as to minimize the impact on the whole aggregator network if any random aggregator fails. By way of example, if one of the aggregators is particularly powerful in terms of data storage capacity, it may be wise to avoid allowing that powerful aggregator to

handle a high percentage of the data capture requests to lessen the adverse impact in the event of failure by that aggregator.

It should be noted that the selection of an additional aggregator to provide additional fault tolerance protection for stored data may be made by reviewing the load status database, the latency data and/or the processing capability database kept at director 500, without regard to whether the additional aggregator is "local" or "remote." This may very well be the case if the aggregators of the system network are connected through reliable, high speed network connections and the geographical distinction may therefore be less important. In this case, the aggregator that is mostly lightly loaded may well be selected to capture redundant data.

In one embodiment, a director may be co-located with the router that routes the traffic to the aggregators, or it may be implemented separately from the router. It should be kept in mind that although Figure 3 shows a director 500 for each of the web server stage, the business logic stage, and the data base stage, there is no requirement that there must be a director for each stage. The provision of a single director may be made for all servers, or some servers may do without a director and dump data directly to an aggregator computer.

BUSINESS LOGIC SERVER

Figure 6 illustrates an exemplary business logic server 600 having a business code module 602, a filtering/policy module 604, a handler module 606 and an event processor 608. The information that is to be used for data mining is collected by the servers by a procedure termed "raising an event". For example, the process of a user logging in through a web server may raise a number of events such as the login name, the IP address the user is logging in from, and whether the login is successful. Code module 602 provides the code that drives the functions of server 600. Code module 602 may represent any suitable business application software that is running on server 600. This software is the focus of the data collection efforts for data mining. Within this code, raise event calls are placed to signal when the event does occur, i.e., is raised. Therefore, code module 602 may contain a number of raise event calls as shown, which allow notice to be given when that an event occurs so that the data may be collected.

Use of a raise event call is well known in the art. Advantageously, raise event calls may be sprinkled liberally throughout the code, wherever it is possible that data might need to be collected.

Handler module 606 includes a number of handler subroutines, for example, one for each event to be raised. There may also be instances when a handler is created to accommodate more than one event or, alternatively, where the occurrence of an event causes more than one handler to be invoked. There may also be instances where handlers are not created for certain events, such as when be events are raised but the information generated by the occurrence of these events do not need to be collected at this time. Each handler subroutine, tailored to its corresponding event, is invoked when two conditions are met, namely, when an event corresponding to that handler occurs and if that particular event had previously been chosen to generate information for future data collection as explained in Figure 9. The process of checking whether both conditions have been met is included among the functions of event processor 608 and is described in Figure 9. Moreover, if a handler subroutine is invoked, that subroutine runs and may perform many tasks, for example, capturing relevant data to the aggregator for collection. Event processor 608 may be any suitable software module for implementing Figures 9 and 10.

Filtering module 604 includes a conceptual event table that lists all the events that have been created in code module 602 and can be raised when the user activity corresponding to that event occurs. The events in this event table may be turned on or off, depending on the purpose, for example, if the task at hand is to monitor security, the security-related events would be turned on. The same would hold true for other tasks such as load balancing, cross-selling, dynamic decision making, etc. Events may be turned on or off through the use of a simple flag for each event or other similar technique. The filtering module may also be provided with logic to enable the module to make certain strategic decisions based on feedback received. An exemplary process by which an exemplary filtering module operates is provided in the flowcharts that will be described later in Figures 9 and 10.

FLOW DIAGRAMS FOR EVENT PROCESSING

Figure 7 illustrates a process flow 700 of creating an event for use in data mining. In 704, a user activity is examined to determine if it is a candidate for data

mining, that is, if that particular type of user activity is suitable for data collection. This is achieved by having the programmer analyze the primary business application module which he is writing. If a user activity is deemed to be a suitable candidate for data mining, an event having a unique label is created for that user activity in 710, followed by the documenting of the existence of that user activity event by recordation of its name in 714. For example, if a login involves ten steps, some of these steps might be selected to create events. By way of example, some events that may be created during the login process may include bi-state events such as login success or failure, or singular events such as user name, login time, IP address, etc. Other types of events may relate to specific business applications such as human resources, accounting, budgeting, planning, etc. Events related to human resources include: employee enrollment, time card entry, vacation request, benefits administration, etc.

Documentation of each event is done independently and is not part of the code. In 718, raise event calls are placed in appropriate locations in the code so that when the user activity occurs, that event will be raised, and data relating to that event can be collected. In 724, the code is compiled to store the selected events which will allow the establishment of linkage to listeners who subscribe to specified events. The process of compiling the code creates calls into operating system code that will respond to a raise event call at run time with the proper action which involves notifying registered listeners. Such linkage into the operating system to connect with a listener is known in the art.

Since the created events can be dynamically turned on or off without any need to modify or recompile the code, the programmer can be opportunistic and place raise event calls in every location where data that may be remotely interesting is generated, and the programmer is only required to make judgment calls to eliminate repetitive or redundant data gathering. This would eliminate the risk of omitting what might later be considered as important data without raising concerns of ending up with an overabundance of unnecessary data. Advantageously, the events are turned on and off dynamically using the event table of module 604.

Figure 8 illustrates a process flow 800 of setting up a listener for capturing a selected event by that listener. The process begins with an expert reviewing the list of documented events in 804. Then, based on his knowledge of each domain, the expert

selects an event that is appropriate for a particular business need in 810. At this point in time, the event may be turned on via its flag in the event table of module 604. Advantageously, the event may be toggled on or off at any future time through the event table, which decouples data collection from the actual business application software. In 814, handler code for the selected event is written to handle that event should the selected event be raised. The function of the handler code is primarily to respond to the raise event call and capture the data generated for later storage into an aggregator. It is also conceivable, however, that the handler may do some dynamic processing, though that may be limited by the capability of the CPU of the server. It is presumed that this process flow 800 may be repeated as needed, for example, until a listener has been set up for all the events that the expert has selected, or until handler modules have been created for all the events that may potentially be of interest to present and future subscribing listeners. Setting up a listener and writing handler code is a known technique to those of skill in the art.

Figure 9 illustrates a process flow 900 of processing an event once that event is raised. Process flow 900 is controlled by the event processor 608 and begins when an event is raised in 904. Upon the raising of an event, a daemon thread is signaled as in 906, followed by the obtaining of an event identifier as in 908. Then, in 910, the event is looked up in the event table of filtering module 604 to determine if the event has been turned on, that is, if it has been chosen for data collection. There may also be other qualifiers besides a bi-state decision in the event table to determine whether data generated by the occurrence of a particular event will be raised.

Alternatively, the other qualifiers may be incorporated in the policy module, and a simplified event table limited to a bi-state qualifier selection may be used for this event processing procedure. The decision is made in 914, wherein if the raised event has been turned on, the handler for that event is invoked in 916. Once the handler is invoked, it runs as in 918 and captures data to the aggregator as in 920 by transferring the data over a network connection from the server. However, the handler functions may also be expanded to include utilities other than capturing data such as online data mining in response to the occurrence of a corresponding event. After capturing the desired information, the daemon thread is suspended as in 922. If the event has not been turned on, the process flow proceeds from decision point 914

directly to 922 to suspend the daemon thread. Once the daemon thread is suspended, it waits for the next time it is signaled, for example, when another event is raised.

The use of daemon threads in processing events allows for asynchronous performance of the event processor. When an event is raised a daemon thread takes over the task of determining whether the event is turned on, and if so, it invokes the corresponding handler for the event to perform handler functions such capturing data, dumping the data to an aggregator, etc. Meanwhile, the code that raised the event proceeds with its path to perform its primary business logic functions whereas if the event processor were designed as a synchronous machine, it would not be possible to achieve optimal performance by using multiple processors that run in parallel. If there is only a single processor, however, it would perform as a synchronous machine without the capability to maximize performance, and the users would not be able to take advantage of the asynchronous characteristics of this inventive infrastructure.

Figure 10 illustrates a process flow 1000 to filter the process of collecting data by taking actions in response to events raised. This process may be used to discover irregularities in the use of the software or correlation amongst events. The filtering process is directed by the filtering/policy module 604, which includes the event table listing all the events that can be raised as well as whether they are on or off. The events may be toggled on and off manually, or the toggling process may be accomplished by automating the code. Filtering process 1000 provides an example of how such code automation may be accomplished. The process begins at 1004 when notification from the event processor signals that an event has been raised or that a handler has been invoked and is running in response to the event being raised. A counter specific to each type of event is provided for implementation, and this counter is incremented each time an event of that type occurs in 1010. If the count threshold for a particular event type is exceeded, as determined in 1014, some intervening acts may take place, for example, some events may be toggled on or off as in 1016, and the counters may be reset as in 1018. If the count threshold has not been exceeded, process flow 1000 comes to a halt as shown in 1022.

By way of example, suppose the event processor sends a number of notifications of unsuccessful login events, which may possibly be some unscrupulous users trying to use illegal passwords to break into the system. If the implemented counter has a threshold of five login failures, once there are five consecutive login

failures the event for login IP address is toggled on and the machine starts to store the IP addresses that all subsequent logins originate from in order to capture the addresses of the unscrupulous users. At this point, the counter for login failures may be reset. Other unusual events that may be counted include repeated access failures, sequential scanning of large numbers of employee records, etc. Other response mechanisms in the filtering module other than toggling events on and off may include, by way of example, blocking logins from a suspect IP address, sending warning messages to alert responsible personnel such as a systems administrator about security risks, etc.

The improved data collection techniques have many inherent advantages which solve many of the problems encountered with earlier methods, for example, shortage in storage space, processing bottlenecks, and inflexibility in the selection of data to be collected. An important advantage is that the improved approach allows the business logic and the data collection to operate independently of one another. The improved data collection techniques also allows for flexibility in selecting what data is to be collected, usage of aggregators to isolate storage capability so that the data collection process does not affect the regular functions of the servers, and expansion capabilities in terms of data storage, among other advantages.

COMPUTER SYSTEM EMBODIMENT

Figures 11 and 12 illustrate a computer system 900 suitable for implementing embodiments of the present invention. Figure 11 shows one possible physical form of the computer system. Of course, the computer system may have many physical forms ranging from an integrated circuit, a printed circuit board and a small handheld device up to a huge super computer. Computer system 900 includes a monitor 902, a display 904, a housing 906, a disk drive 908, a keyboard 910 and a mouse 912. Disk 914 is a computer-readable medium used to transfer data to and from computer system 900.

Figure 12 is an example of a block diagram for computer system 900. Attached to system bus 920 are a wide variety of subsystems. Processor(s) 922 (also referred to as central processing units, or CPUs) are coupled to storage devices including memory 924. Memory 924 includes random access memory (RAM) and read-only memory (ROM). As is well known in the art, ROM acts to transfer data and instructions uni-directionally to the CPU and RAM is used typically to transfer

data and instructions in a bi-directional manner. Both of these types of memories may include any suitable of the computer-readable media described below. A fixed disk 926 is also coupled bi-directionally to CPU 922; it provides additional data storage capacity and may also include any of the computer-readable media described below. Fixed disk 926 may be used to store programs, data and the like and is typically a secondary storage medium (such as a hard disk) that is slower than primary storage. It will be appreciated that the information retained within fixed disk 926, may, in appropriate cases, be incorporated in standard fashion as virtual memory in memory 924. Removable disk 914 may take the form of any of the computer-readable media described below.

CPU 922 is also coupled to a variety of input/output devices such as display 904, keyboard 910, mouse 912 and speakers 930. In general, an input/output device may be any of: video displays, track balls, mice, keyboards, microphones, touch-sensitive displays, transducer card readers, magnetic or paper tape readers, tablets, styluses, voice or handwriting recognizers, biometrics readers, or other computers. CPU 922 optionally may be coupled to another computer or telecommunications network using network interface 940. With such a network interface, it is contemplated that the CPU might receive information from the network, or might output information to the network in the course of performing the above-described method steps. Furthermore, method embodiments of the present invention may execute solely upon CPU 922 or may execute over a network such as the Internet in conjunction with a remote CPU that shares a portion of the processing.

In addition, embodiments of the present invention further relate to computer storage products with a computer-readable medium that have computer code thereon for performing various computer-implemented operations. The media and computer code may be those specially designed and constructed for the purposes of the present invention, or they may be of the kind well known and available to those having skill in the computer software arts. Examples of computer-readable media include, but are not limited to: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROMs and holographic devices; magneto-optical media such as floptical disks; and hardware devices that are specially configured to store and execute program code, such as application-specific integrated circuits (ASICs),

programmable logic devices (PLDs) and ROM and RAM devices. Examples of computer code include machine code, such as produced by a compiler, and files containing higher level code that are executed by a computer using an interpreter.

Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced within the scope of the appended claims. For example, the computer infrastructure is by no means limited to what is described, in fact, it may include servers that also generate data for data mining but are functionally different from the servers provided in the above examples. Also, the aggregator network may have a wide variety of topologies, for example, the aggregators at a certain level may be linked to multiple aggregator intelligent director. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.

CLAIMS

1. An aggregator intelligent director computer for coordinating data capture requests, said computer comprising:

a processing capability database that indicates the processing capabilities of a plurality of aggregator computers in communication with a server computer, said aggregator computers arranged to store data captured from said server computer;

a load status database that indicates the load present on said aggregator computers;

an aggregator request input line originating from said server computer arranged to request a suitable aggregator computer from said director computer;

an aggregator selection output line destined for said server computer arranged to indicate a suitable aggregator computer to be used for data capture; and

an aggregator selector software module arranged to receive said aggregator request input line and to output an aggregator selection on said aggregator selection output line based upon a selection algorithm that takes into account said processing capability database and said load status database, whereby said director computer coordinates the selection of an aggregator computer.

2. An aggregator intelligent director computer as recited in claim 1 further comprising:

a usage history database that indicates the usage history of said aggregator computers, whereby said selection algorithm may take into account said usage history database.

3. An aggregator intelligent director computer as recited in claim 1 further comprising:

an input feedback line received from said aggregator computers for updating said processing capability database and said load status database dynamically.

4. An aggregator intelligent director computer as recited in claim 1 wherein said director computer is co-located with said server computer.

5. A computer network arranged to capture data comprising:

a plurality of web server computers;

a plurality of business logic server computers;

an aggregator intelligent director computer connected to said web server computers or said business logic server computers, said director computer for coordinating data capture requests from said servers and including an aggregator selector software module arranged to receive an aggregator request and to output an aggregator selection based upon a selection algorithm; and

a network of aggregator computers connected to said director computer and arranged to receive captured data from said servers based upon said aggregator selection from said director computer, whereby said director computer coordinates the selection of an aggregator computer for said servers.

6. A computer network as recited in claim 5 wherein said director computer is co-located with one of said server computers.

7. A method for capturing data from a server computer comprising:

receiving a request to route captured data from said server computer;

checking a database relating to a plurality of data aggregator computers to determine a suitable aggregator computer to be selected;

selecting an aggregator computer using a heuristic based in part upon said database checking; and

sending an identifier of said aggregator computer to said server computer whereby said server computer may send captured data to said aggregator computer.

8. A method for capturing data as recited in claim 7 further comprising:

accepting feedback from said aggregator computers indicating their status; and

including said feedback in said database.

9. A method for capturing data as recited in claim 7 further comprising:

receiving captured data from said server computer; and

routing said captured data to the aggregator computer selected.

10. A method for capturing data as recited in claim 7 wherein said captured data from said server computer originates from raised events in a business application, and wherein said captured data is appropriate for data mining.

11. A method for capturing data as recited in claim 7 further comprising:

checking a processing capability database and a load status database relating to said data aggregator computers to determine a suitable aggregator computer to be selected.

12. A method for capturing data as recited in claim 7 further comprising:

checking a plurality of other databases relating to said data aggregator computers to determine a suitable aggregator computer to be selected.

13. A method of creating a dynamic event for a user activity for use in data capture, said method comprising:

determining that a user activity occurring in a business application program is useful for data capture;

creating a software event for said user activity, said event to be raised when said user activity occurs while said business application program runs;

placing calls to raise said event in the code of said business application program, whereby said event is raised when said user activity occurs;

creating an event table including the name of said event and a flag indicating whether said event is turned on or turned off; and

storing said event table in a software module separate from said business application program, whereby said event may be toggled dynamically in said event table separately from said business application program.

14. A method as recited in claim 13 further comprising:

creating a plurality of events as recited in claim 13, each of said events having an event name and a corresponding flag in said event table;

selecting at least one of said events that is suitable for data capture for a particular business need; and

writing handler code for each selected event, said handler code being arranged to handle said selected event should said selected event be raised.

15. A method as recited in claim 13 wherein said software module is a policy module that dictates a business policy for capturing data through the use of said event table.

16. A method as recited in claim 13 wherein said data capture is for the purpose of data mining, said method further comprising:

creating software events for a plurality of user activities that are unrelated to a particular business need;

placing calls to raise said events in the code of said business application program, said calls being placed without regard to a particular strategy for collecting data;

storing the names of said events along with corresponding flags in said event table of said software module; and

editing said flags of said events in said event table to turn on data capture for certain events related to a particular business need, whereby data mining may be used on data captured from said certain events.

17. A method of processing a dynamic event, said method comprising:

receiving an indication of an event being raised in a business application program on a computer, said event being related to a user activity that has occurred;

obtaining an event identifier of said raised event;

locating an event table that includes a list of event identifiers along with corresponding flags indicating whether said events are turned on or are turned off;

looking up said event identifier in said event table to determine if said raised event is turned on or turned off;

wherein if said raised event is turned on, the method further comprises invoking a handler for said raised event to process said raised event, whereby events are handled in a dynamic fashion.

18. A method as recited in claim 17 further comprising:

delivering data from said raised event to an aggregator computer for storage.

19. A method as recited in claim 17 further comprising:

wherein if said raised event is turned off, the method further comprises suspending the handling of said raised event, whereby no data is captured corresponding to said raised event.

20. A method as recited in claim 17 wherein said event table is stored in a software policy that dictates a business policy for capturing data through the use of said event table.

21. A method as recited in claim 17 wherein said data capture is for the purpose of data mining, said method further comprising:

editing said flags of said events in said event table to turn on data capture for certain events related to a particular business need, whereby data mining may be used on data captured from said certain events.

22. A server computer for use in data capture comprising:

a software code module representing a business application program, said code module including a plurality of calls to raise events that are associated with user activities;

a handler module including a plurality of software handlers, each handler being arranged to process one of said raised events;

an event table including the name of each of said events and a flag indicating whether each event is turned on or turned off; and

an event processor module arranged to respond to said raised events and to instruct said handler module based upon said event table, whereby by raised events are handled in a dynamic fashion.

23. A server computer as recited in claim 22 wherein said event table is located in a software policy module that dictates a business policy for capturing data through the use of said event table.

24. A server computer as recited in claim 22 wherein said data capture is for the purpose of data mining, said server including:

an aggregator database for storage of captured data, whereby data mining may be used on data captured from said events.

25. A server computer as recited in claim 22 wherein said server computer is a business application server that is distinct from an aggregator computer used to capture data.

26. A method of implementing a business policy for use in data capture, said method comprising:

creating a software event for a user activity that occurs during a business application program, said event to be raised when said user activity occurs while said business application program runs;

placing calls to raise said event in the code of said business application program, whereby said event is raised when said user activity occurs;

creating an event table including the name of said event, a flag indicating whether said event is turned on or turned off and a counter specific to said event; and

storing said event table in a software module separate from said business application program, whereby said counter associated with said event may be used to implement said policy for data capture from said business application program.

27. A method as recited in claim 26 further comprising:

creating a plurality of events as recited in claim 26, each of said events having an event name, a corresponding flag and a counter specific to each event in said event table;

selecting at least one of said events that is suitable for data capture for a particular business need; and

writing handler code for each selected event, said handler code being arranged to handle said selected event should said selected event be raised.

28. A method as recited in claim 26 wherein said software module is a policy module that dictates a business policy for capturing data through the use of said counter of said event table.

29. A method as recited in claim 26 wherein said data capture is for the purpose of data mining, said method further comprising:

creating software events for a plurality of user activities that are unrelated to a particular business need;

placing calls to raise said events in the code of said business application program, said calls being placed without regard to a particular strategy for collecting data;

storing the names of said events along with corresponding flags and corresponding counters in said event table of said software module; and

editing thresholds corresponding to said counters of said events in said event table to assist in implementing data capture for a particular business need, whereby data mining may be used on data captured from said certain events.

30. A method of implementing a data capture policy using dynamic events, said method comprising:

receiving an indication of an event being raised in a business application program on a computer, said event being related to a user activity that has occurred;

obtaining an event identifier of said raised event;

locating an event table that includes a list of event identifiers along with a counter for each event and a count threshold for each event;

incrementing the counter for said raised event to indicate that said raised event has occurred again;

determining whether the counter for said raised event has exceeded the count threshold for said raised event;

wherein if said count threshold has been exceeded, the method further comprises taking a particular action in response to said count threshold being exceeded, whereby a particular data capture policy may be implemented.

31. A method as recited in claim 30 further comprising:

delivering data from said raised event to an aggregator computer for storage if it is determined that a flag associated with said raised event in said event table is turned on.

32. A method as recited in claim 30 further comprising:

wherein if said count threshold is not exceeded, the method further comprises handling said raised event normally, whereby no change is made corresponding to said raised event.

33. A method as recited in claim 30 wherein said event table is stored in a software policy that dictates a business policy for capturing data through the use of said event table.

34. A method as recited in claim 30 wherein said data capture is for the purpose of data mining, said method further comprising:

editing said count thresholds of said events in said event table to modify said data capture policy for a particular business need, whereby data mining may be used on data captured from said certain events.

35. A method as recited in claim 30 wherein said taking a particular action includes toggling flags of selected events, modifying count thresholds or resetting counters of said events, whereby a data capture policy may be implemented in response to said counter exceeding said count threshold.

36. A server computer for use in data capture comprising:

a software code module representing a business application program, said code module including a plurality of calls to raise events that are associated with user activities;

a handler module including a plurality of software handlers, each handler being arranged to process one of said raised events;

an event table including the name of each of said events, a flag indicating whether each event is turned on or turned off and an event counter indicating how many times each event has occurred; and

an event processor module arranged to respond to said raised events and to instruct said handler module based upon said event table, whereby a data capture policy is dictated by said event table.

37. A server computer as recited in claim 36 wherein said event table is located in a software policy module that dictates a business policy for capturing data through the use of said event table.

38. A server computer as recited in claim 36 wherein said data capture is for the purpose of data mining, said server including:

an aggregator database for storage of captured data, whereby data mining may be used on data captured from said events.

39. A server computer as recited in claim 36 wherein said server computer is a business application server that is distinct from an aggregator computer used to capture data.

40. A server computer as recited in claim 36 further including:

a count threshold associated with each event in said event table, whereby a comparison may be made between said event counters and said count thresholds for each of said events in said event table and a data capture policy implemented.

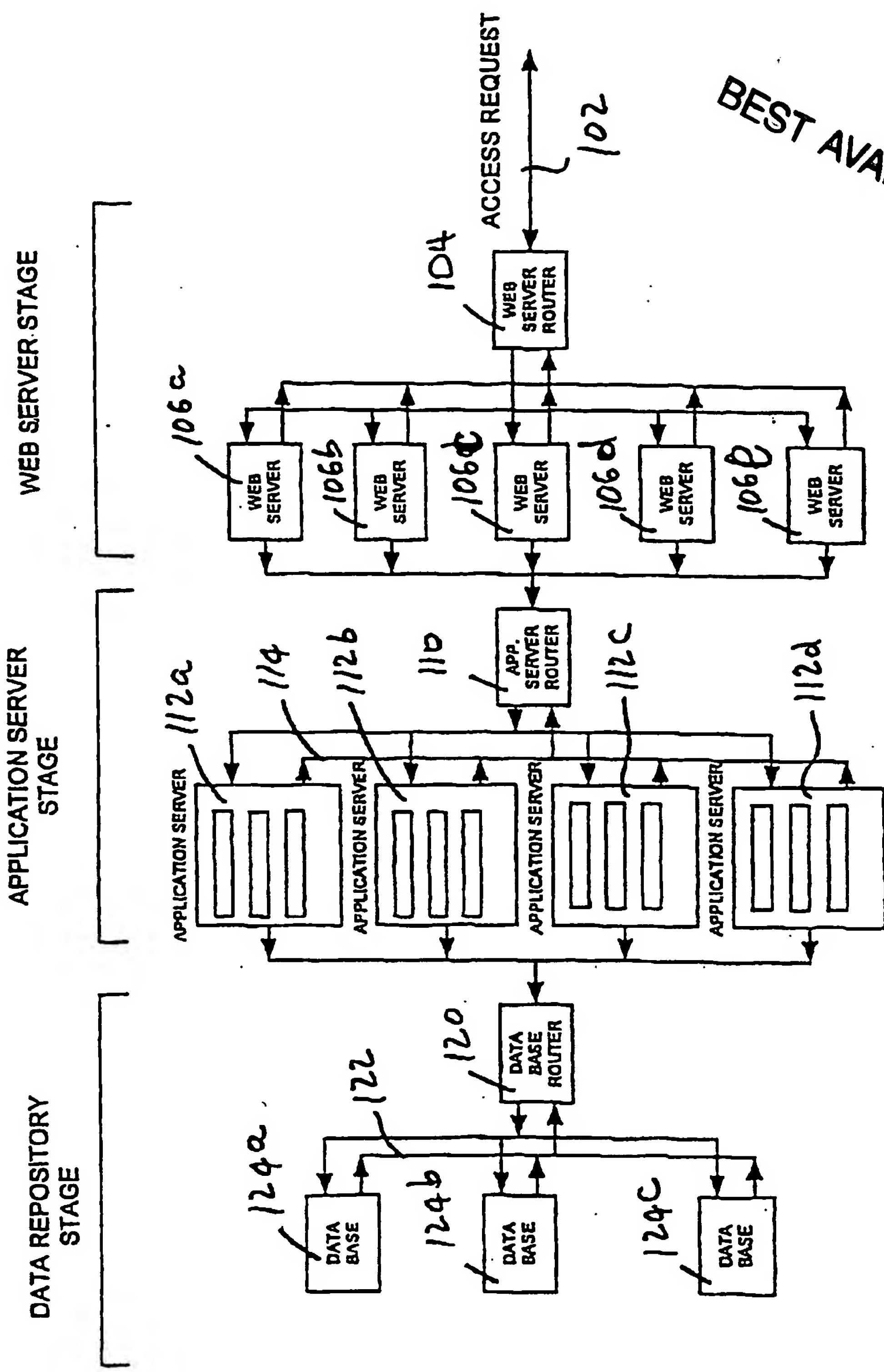
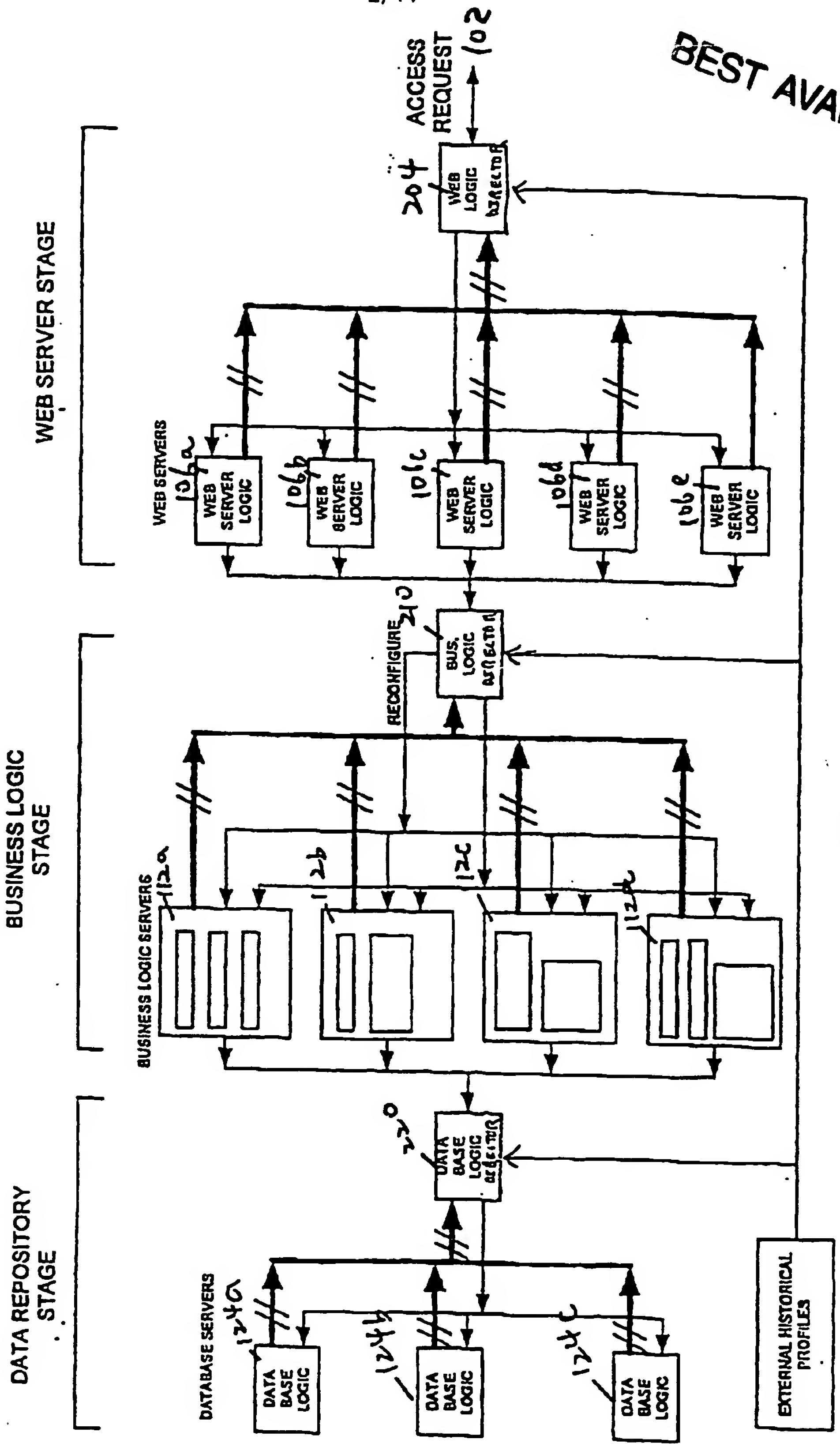


FIG. 1 (Prior art)



ACCESS REQUEST
102
204
WEB LOGIC DIRECTOR
WEB SERVER STAGE
WEB SERVERS
106a
106b
106c
106d
106e
WEB SERVER LOGIC
210
RECONFIGURE
BUS. LOGIC DIRECTOR
BUSINESS LOGIC STAGE
BUSINESS LOGIC SERVERS
112a
112b
112c
112d
220
DATA BASE LOGIC DIRECTOR
DATA REPOSITORY STAGE
DATABASE SERVERS
124a
124b
124c
DATA BASE LOGIC
EXTERNAL HISTORICAL PROFILES

FIG. 2

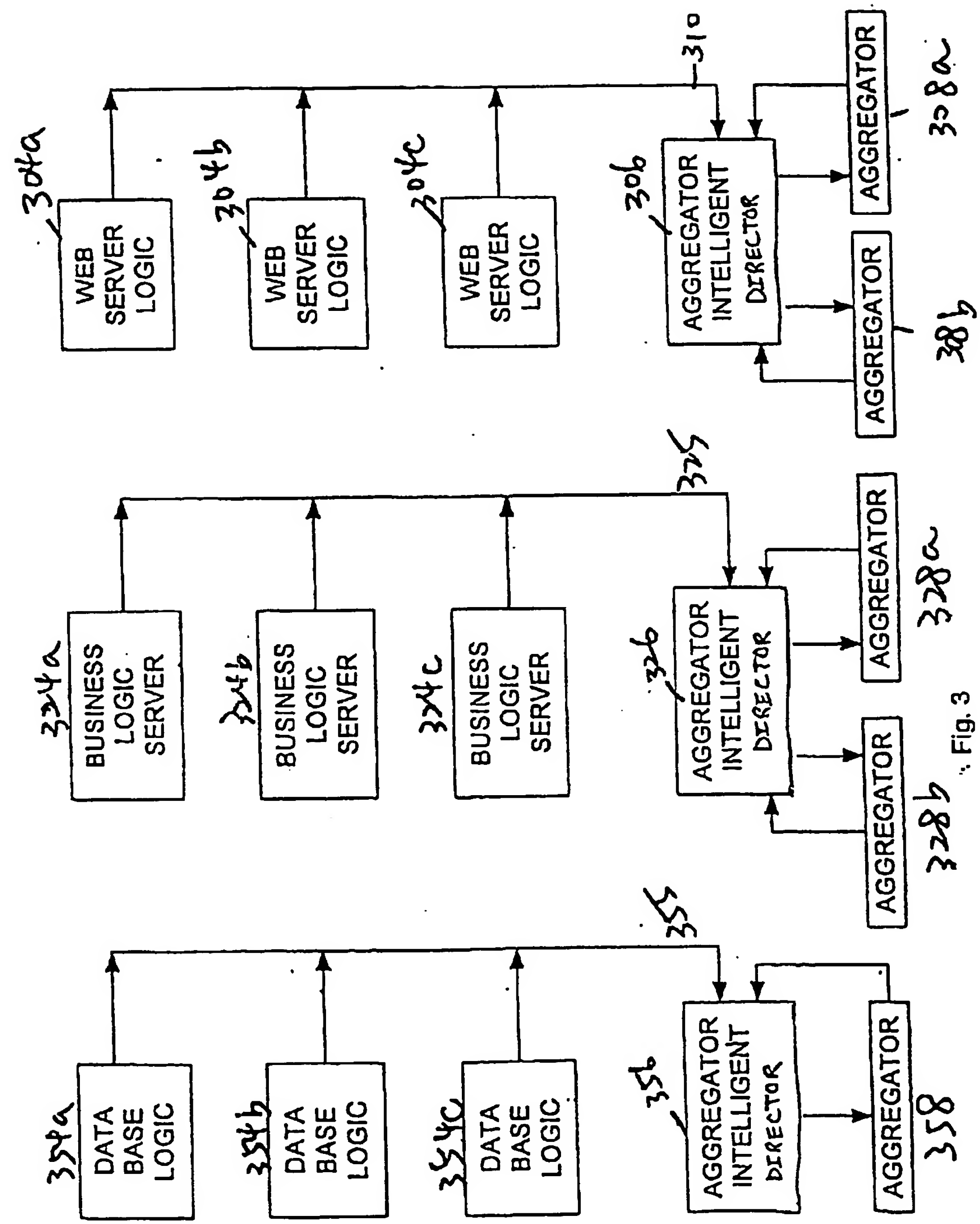

WEB SERVER LOGIC
304a
304b
304c
310
AGGREGATOR INTELLIGENT DIRECTOR
306
AGGREGATOR
308a
308b
BUSINESS LOGIC SERVER
324a
324b
324c
325
326
328a
328b
DATA BASE LOGIC
354a
354b
354c
355
356
358

Fig. 3

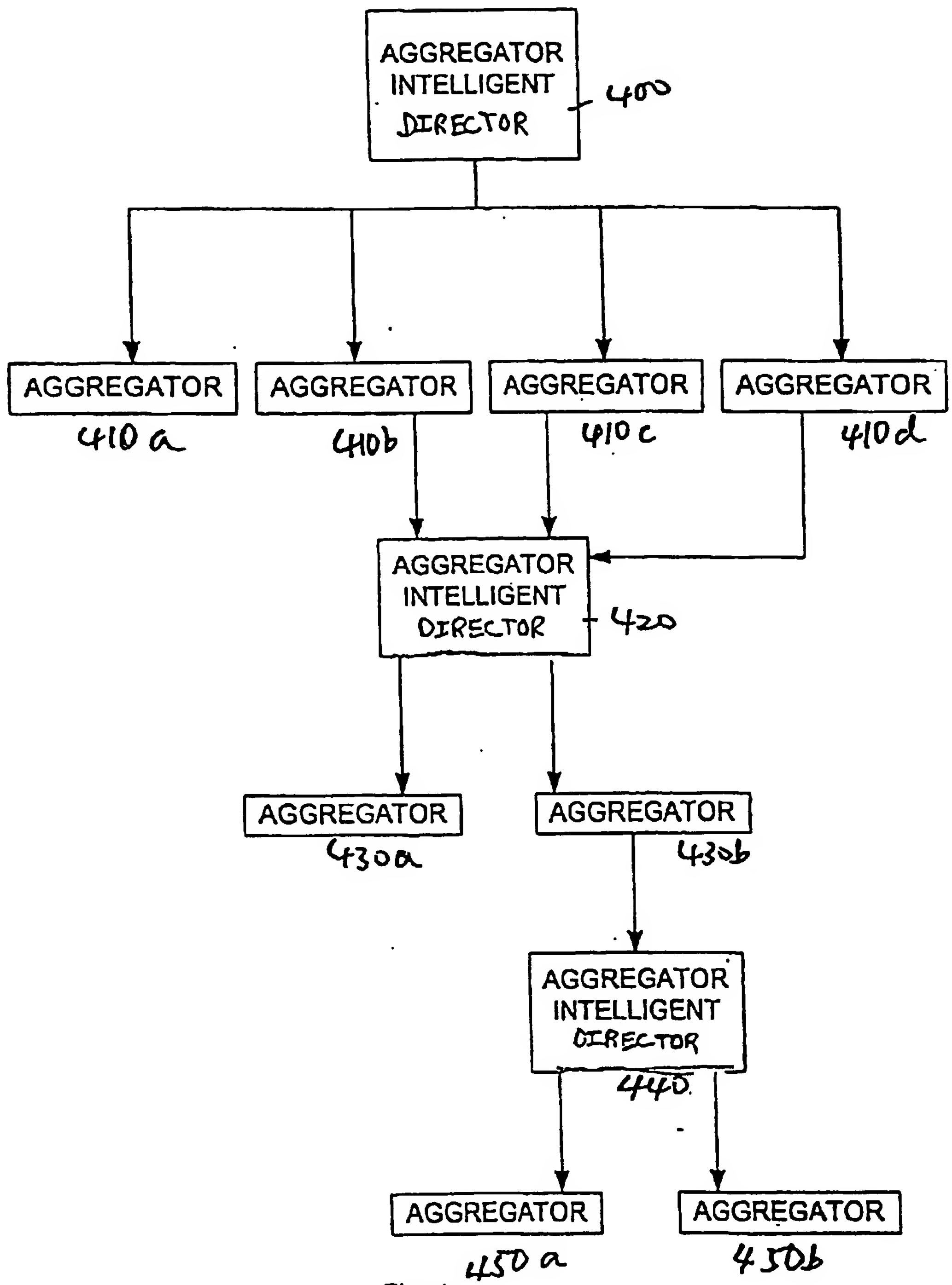

AGGREGATOR INTELLIGENT DIRECTOR
400
AGGREGATOR
410 a
AGGREGATOR
410b
AGGREGATOR
410 c
AGGREGATOR
410d
AGGREGATOR INTELLIGENT DIRECTOR
420
AGGREGATOR
430a
AGGREGATOR
430b
AGGREGATOR INTELLIGENT DIRECTOR
440
AGGREGATOR
450 a
AGGREGATOR
450b

Fig. 4

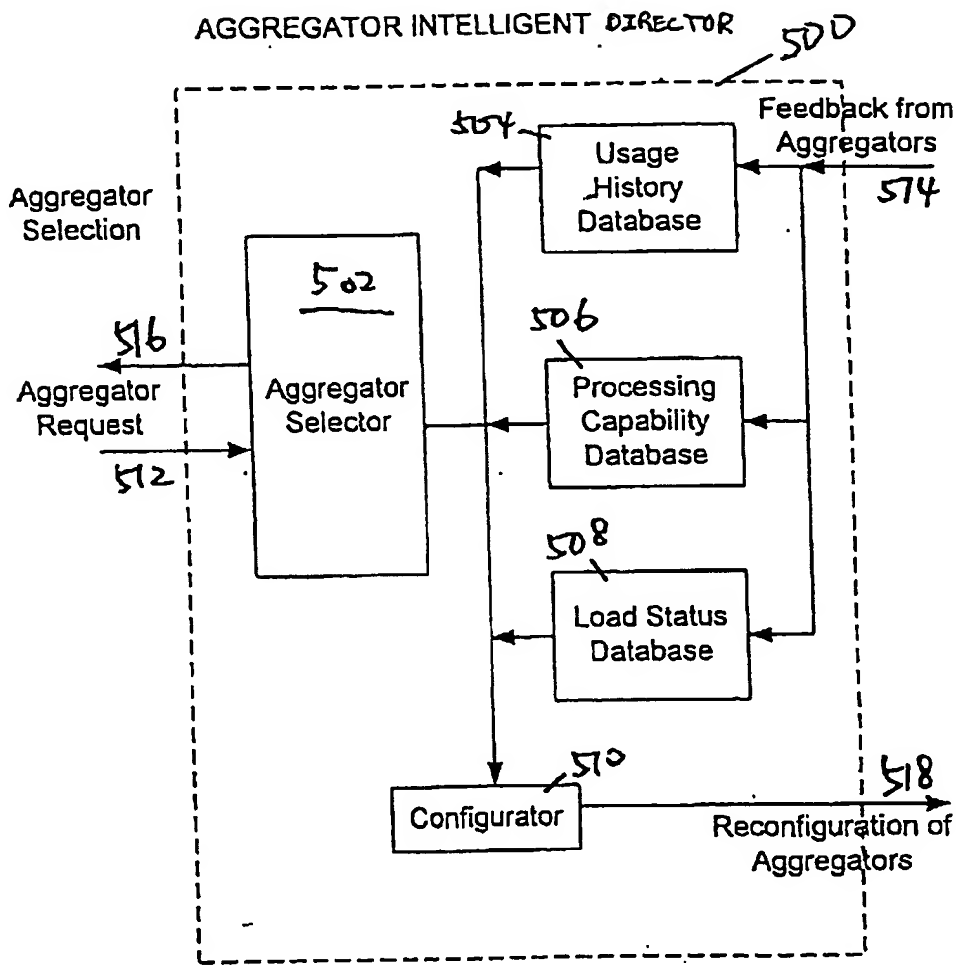

AGGREGATOR INTELLIGENT DIRECTOR
500
Feedback from
Aggregators
514
504
Usage
History
Database
Aggregator
Selection
502
Aggregator
Selector
516
Aggregator
Request
512
506
Processing
Capability
Database
508
Load Status
Database
510
Configurator
518
Reconfiguration of
Aggregators

Fig. 5

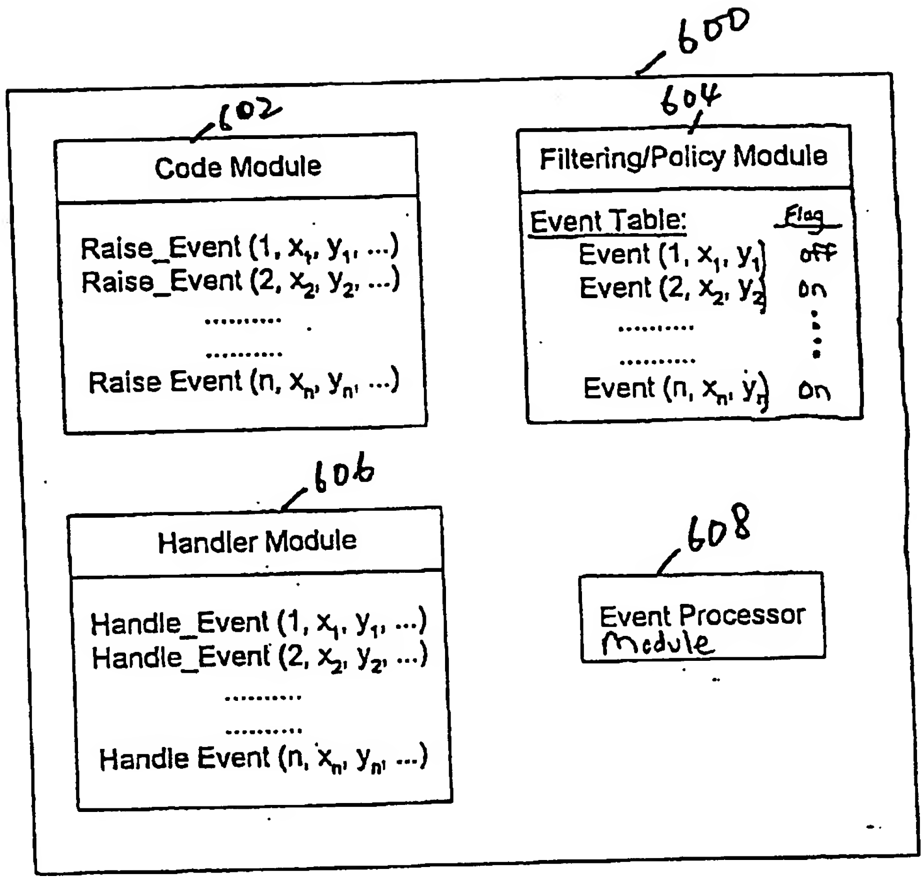
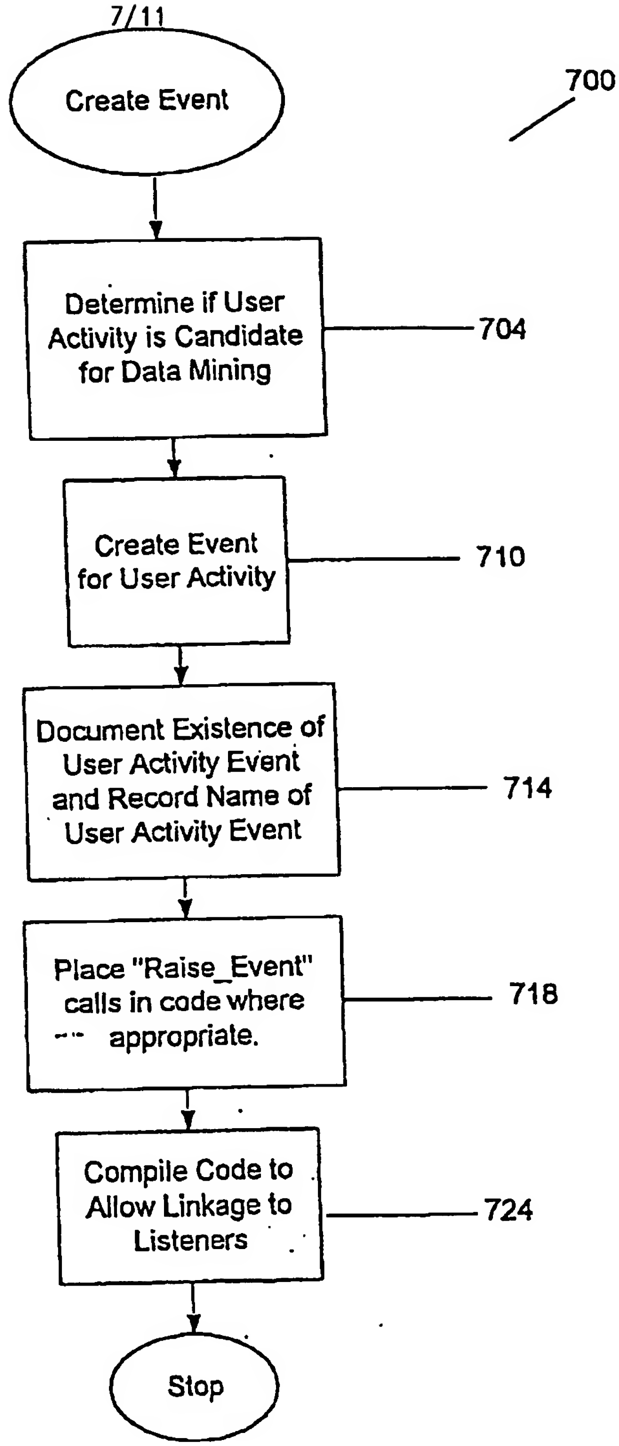

600
602
Code Module
Raise_Event (1, x1, y1, ...)
Raise_Event (2, x2, y2, ...)
....
....
Raise Event (n, xn, yn, ...)
604
Filtering/Policy Module
Event Table:
Flag
Event (1, x1, y1) off
Event (2, x2, y2) on
....
....
Event (n, xn, yn) on
606
Handler Module
Handle_Event (1, x1, y1, ...)
Handle_Event (2, x2, y2, ...)
....
....
Handle Event (n, xn, yn, ...)
608
Event Processor Module

Fig. 6



Create Event
700
Determine if User Activity is Candidate for Data Mining
704
Create Event for User Activity
710
Document Existence of User Activity Event and Record Name of User Activity Event
714
Place "Raise_Event" calls in code where appropriate.
718
Compile Code to Allow Linkage to Listeners
724
Stop

Fig. 7

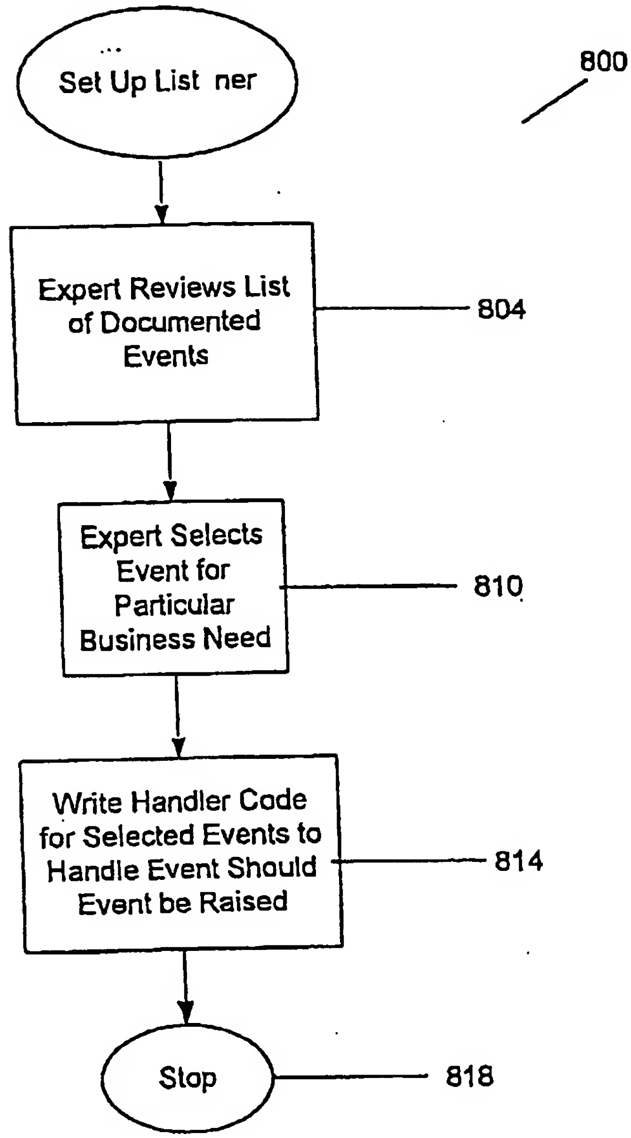

800
Set Up List ner
Expert Reviews List of Documented Events
804
Expert Selects Event for Particular Business Need
810
Write Handler Code for Selected Events to Handle Event Should Event be Raised
814
Stop
818

Fig. 8

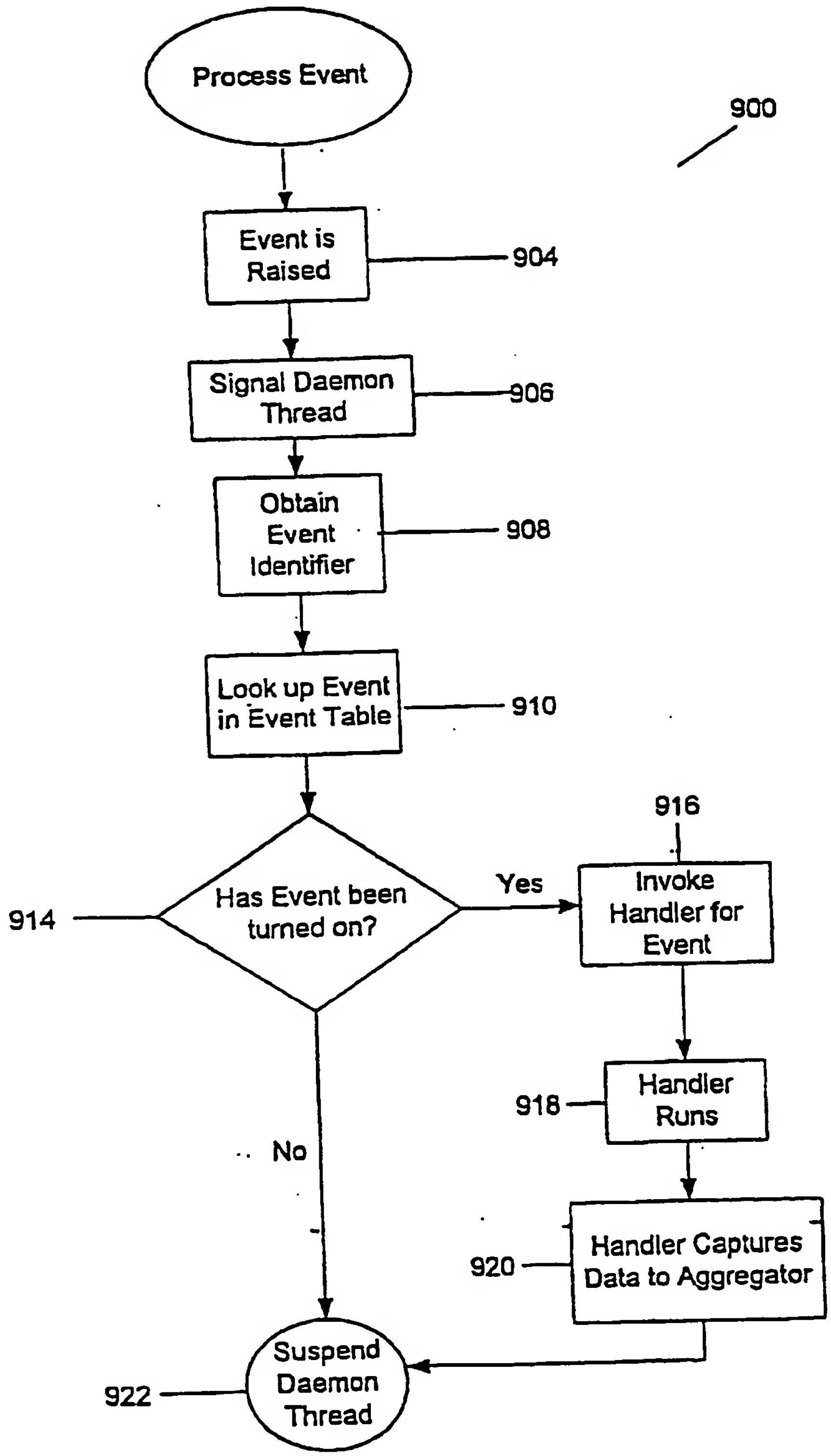

Process Event
900
Event is Raised
904
Signal Daemon Thread
906
Obtain Event Identifier
908
Look up Event in Event Table
910
916
914
Has Event been turned on?
Yes
Invoke Handler for Event
918
Handler Runs
No
920
Handler Captures Data to Aggregator
922
Suspend Daemon Thread

Fig. 9

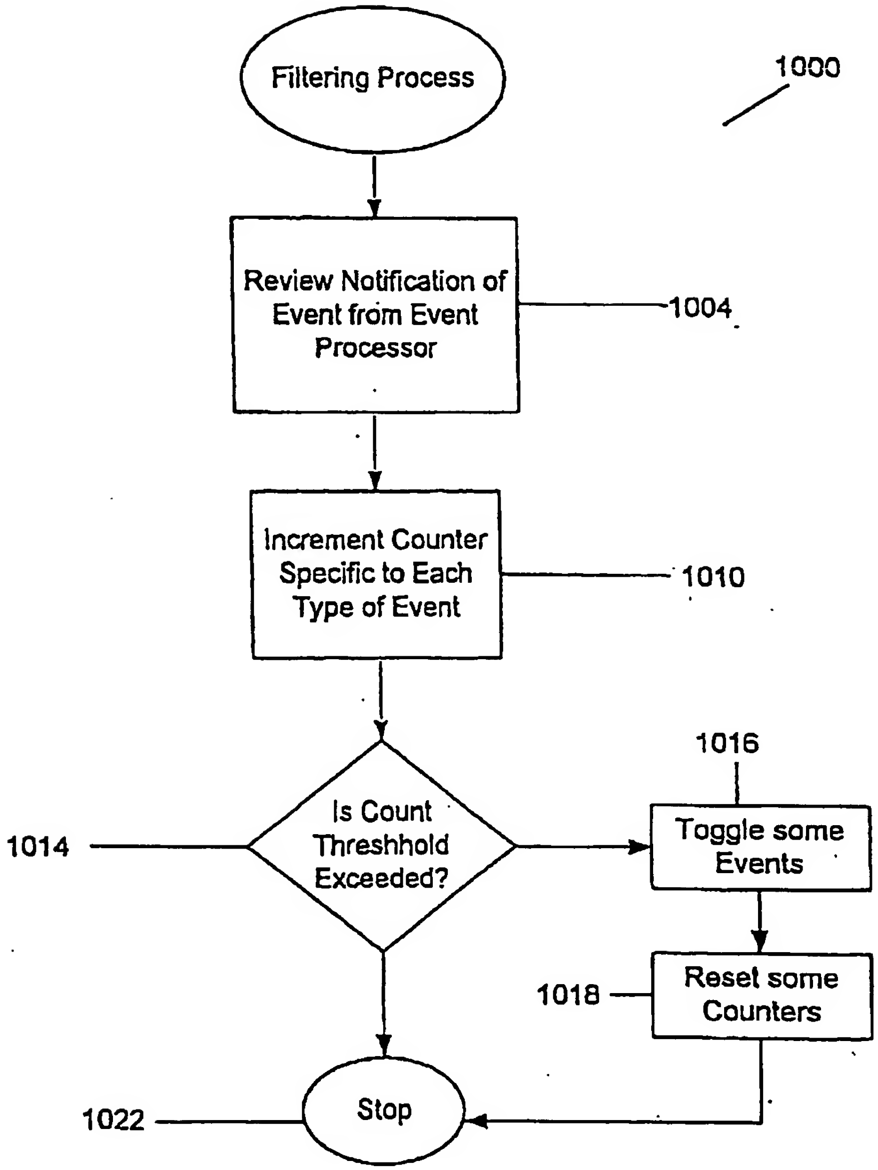

Filtering Process
1000
Review Notification of Event from Event Processor
1004
Increment Counter Specific to Each Type of Event
1010
Is Count Threshhold Exceeded?
1014
1016
Toggle some Events
Reset some Counters
1018
Stop
1022

Fig. 10

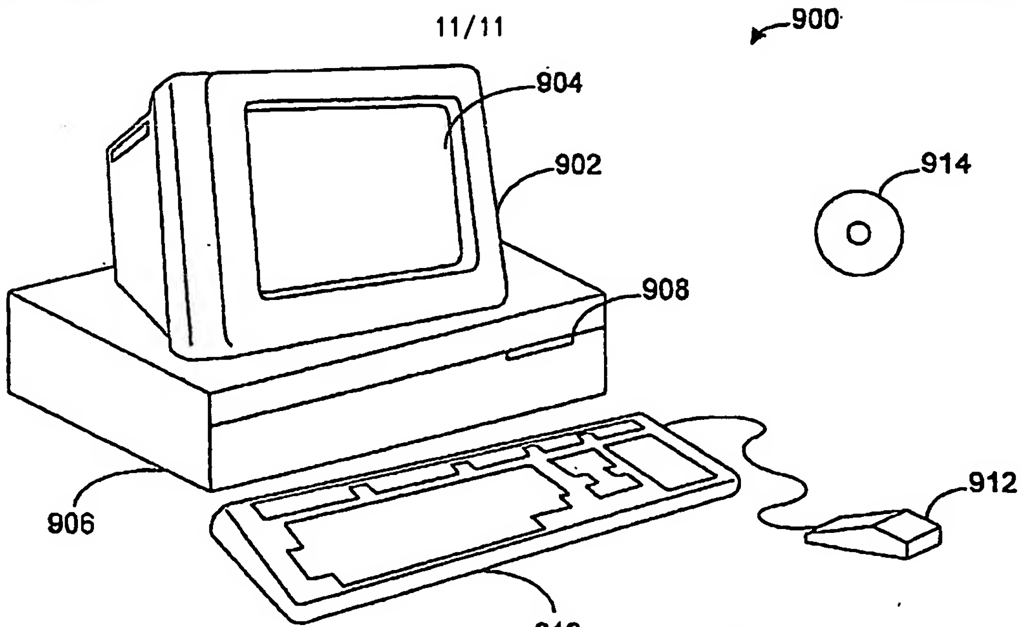

900
904
902
914
908
912
906
910

FIG. 11

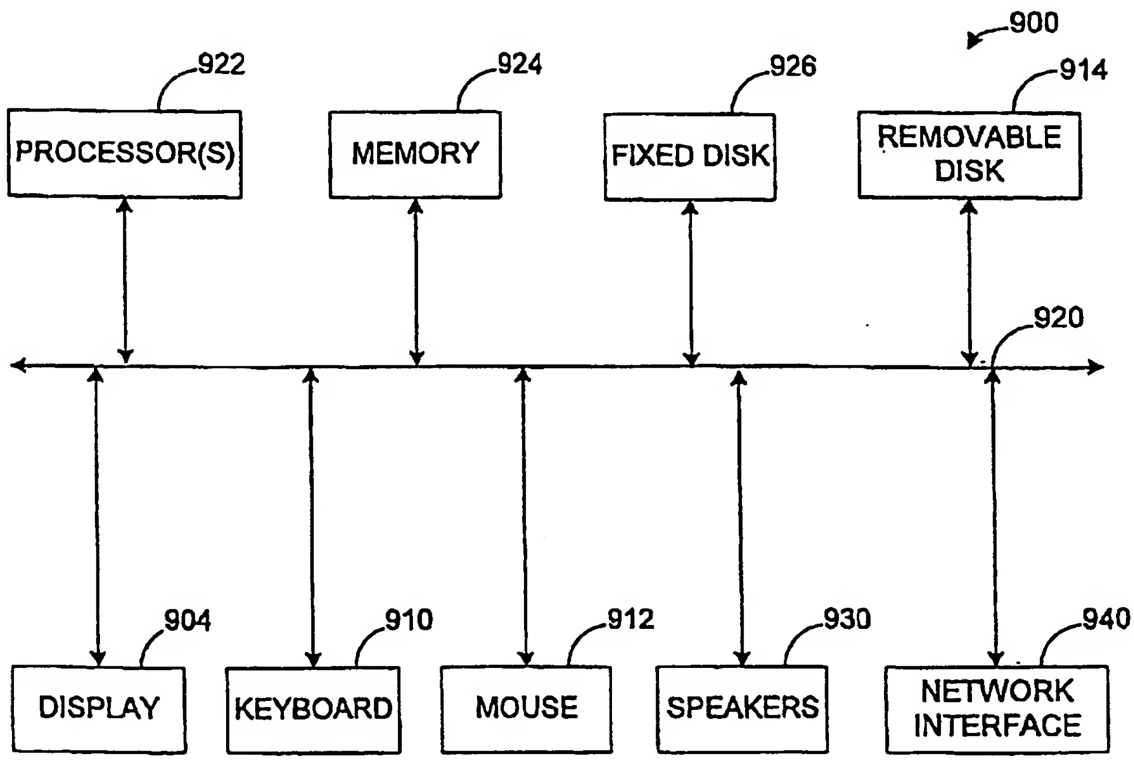

900
922
924
926
914
PROCESSOR(S)
MEMORY
FIXED DISK
REMOVABLE DISK
920
904
910
912
930
940
DISPLAY
KEYBOARD
MOUSE
SPEAKERS
NETWORK INTERFACE

FIG-12

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/18183

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :GO6F 17/30
US CL : 707/1, 3, 4, 5, 10, 102, 104, 200; 345/326, 333, 340, 348, 352.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/1, 3, 4, 5, 10, 102, 104, 200; 345/326, 333, 340, 348, 352.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST, IEEE
Search terms: data mining, server computer, aggregator, web server, business logic.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,446,885 A (MOORE et al) 29 August 1995, col. 3, lines 56-67, col. 4, lines 1-18 and lines 45-62, col. 6, lines 22-67, col. 7, lines 9-28, col. 13, lines 6-67, col. 14, lines 1-6, col. 15, lines 3-64, col. 17, lines 18-68, col. 19, lines 1-68, col, 21, lines 1-20, col. 23, lines 55-68, col. 24, lines 1-4, and col. 30, lines 13-68.	1-40
Y	US 5,890,150 A (USHIJIMA et al) 30 March 1999, col. 2, lines 6-39, col. 3, lines 11-25, lines 41-48, and lines 59-65, col. 5, lines 35-67, and col. 8, lines 11-15.	1-40

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier document published on or after the international filing date	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 13 AUGUST 2000	Date of mailing of the international search report 07 SEP 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer KIN VU Eugenia Logan Telephone No. (703) 305-4393

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/18183

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,873,095 A (GORE) 16 February 1999, col. 1, lines 45-67, col. 2, lines 1-3, lines 36-51, and lines 66-67, col. 3, lines 1-67, col. 4, lines 1-24, col. 5, lines 8-15, col. 6, lines 32-65, and col. 7, lines 7-19.	1-40